(19) JAPANESE PATENT OFFICE

PATENT ABSTRACTS OF JAPAN

(11) Publication number: 07244595 A

(43) Date of publication of application: 19.09.95

(51) Int. Cl G06F 9/46

(21) Application number: 06035535

(22) Date of filing: 07.03.94

(71) Applicant: FUJITSU LTD

(72) Inventor: SHIMIZU TOSHIYUKI
ISHIHATA HIROAKI

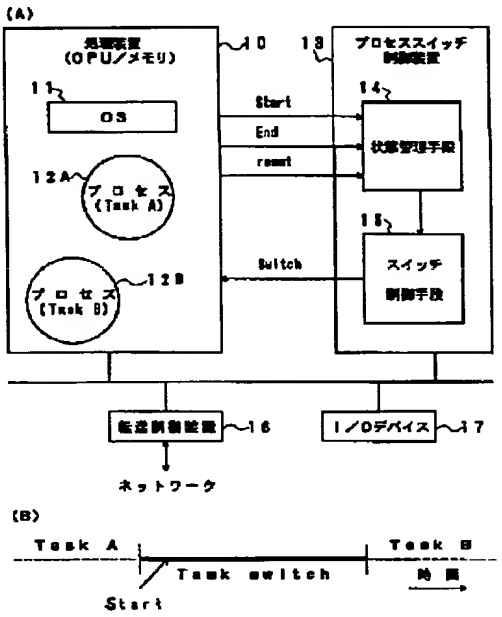
# (54) PROCESS SWITCH CONTROLLER AND PROCESS CONTROLLING METHOD

(57) Abstract:

PURPOSE: To attain highly efficient message transmission and I/O without passing it through an OS by controlling a process switch signal so that a process switch is not turned on in a switch suppressing state.

CONSTITUTION: A state managing means 14 in a process switch controller 13 manages a state including at least a free state capable of turning on the process switch and a switch suppressing state for suppressing the turning-on of the process switch because a process being executed at present is in input/output processing or in message communication. A switch control means 15 provides timing for turning on the process switch with width in accordance with a state, and in the switch suppressing state, controls a process switch signal so as not to turn on the switch. Thereby the sending or I/O of a message can be directly controlled in a user level even in a multi-processing environment, the control can be attained by low overhead and the performance of a computer can be efficiently applied.





(19)日本国特許庁（ＪＰ）　(12) 公開特許公報（Ａ）　(11)特許出願公開番号

特開平7－244595

(43)公開日　平成７年(1995)９月19日

| (51)Int.Cl.[6] | 識別記号 | 庁内整理番号 | Ｆ Ｉ | 技術表示箇所 |
|---|---|---|---|---|
| Ｇ０６Ｆ　9/46 | ３４０　Ｂ | 7737－5B | | |

審査請求　未請求　請求項の数７　ＯＬ　（全 11 頁）

(21)出願番号　特願平6－35535

(22)出願日　平成６年(1994)３月７日

(71)出願人　000005223
富士通株式会社
神奈川県川崎市中原区上小田中1015番地

(72)発明者　清水　俊幸
神奈川県川崎市中原区上小田中1015番地
富士通株式会社内

(72)発明者　石畑　宏明
神奈川県川崎市中原区上小田中1015番地
富士通株式会社内

(74)代理人　弁理士　小笠原　吉義　（外２名）

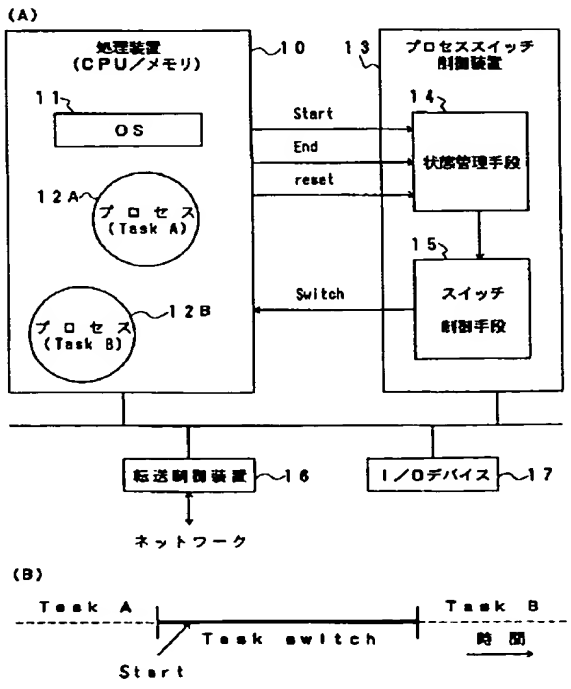
(54)【発明の名称】　プロセススイッチ制御装置およびプロセス制御方法

(57)【要約】

【目的】マルチプロセッシング・システムにおけるプロセススイッチ制御装置およびプロセス制御方法に関し，マルチプロセッシング環境でのユーザレベルの入出力およびメッセージ通信を可能とし，低いオーバーヘッドの入出力およびメッセージ通信を実現することを目的とする。

【構成】プロセススイッチを起こすことのできるフリー状態とクリティカルな状態とを状態管理手段14により管理し，これらの状態に応じてプロセススイッチを起こすタイミングに幅を持たせ，現在実行中のプロセスが入出力中またはメッセージ通信中のクリティカルな状態にある場合には，スイッチ制御手段15によりプロセススイッチが起きないようにプロセスのスイッチ信号を制御する

本発明の原理説明図



1

【特許請求の範囲】

【請求項１】　複数のプロセスが時分割的に動作するマルチプロセッシング・システムにおいて用いられるプロセススイッチ制御装置であって，少なくとも，プロセススイッチを起こすことのできるフリー状態と，現在実行中のプロセスが入出力中またはメッセージ通信中であるためにプロセススイッチを起こすことを抑止するスイッチ抑止状態とを含む状態を管理する状態管理手段(14)と，前記状態に応じてプロセススイッチを起こすタイミングに幅を持たせ，前記スイッチ抑止状態にあるときにはプロセススイッチが起きないようにプロセスのスイッチ信号を制御するスイッチ制御手段(15)とを備えたことを特徴とするプロセススイッチ制御装置。

【請求項２】　複数のプロセスが時分割的に動作するマルチプロセッシング・システムにおいて用いられるプロセススイッチ制御装置であって，少なくとも，プロセススイッチを起こすことのできるフリー状態と，現在実行中のプロセスが入出力中またはメッセージ通信中であるためにプロセススイッチを起こすことを抑止するスイッチ抑止状態と，プロセススイッチのトリガとなるタイマー・イベントが所定の時間内に発生する状態になったことを示すスイッチ予告促進状態とを含む状態を管理する状態管理手段(14)と，前記状態に応じてプロセススイッチを起こすタイミングに幅を持たせ，前記スイッチ抑止状態にあるときにはプロセススイッチが起きないようにプロセスのスイッチ信号を制御するとともに，前記スイッチ予告促進状態において現在実行中のプロセスが入出力中またはメッセージ通信中に入ることを示すイベントが発生したときには，そのプロセスが入出力中またはメッセージ通信中に入る前にプロセスのスイッチ信号を出力するスイッチ制御手段(15)とを備えたことを特徴とするプロセススイッチ制御装置。

【請求項３】　請求項１または請求項２記載のプロセススイッチ制御装置において，前記スイッチ制御手段(15)は，前記スイッチ抑止状態においてはプロセススイッチを遅延させ，現在実行中のプロセスの入出力終了またはメッセージ通信終了のイベントによりプロセスのスイッチ信号を出力し，プロセススイッチを起こす手段を備えたことを特徴とするプロセススイッチ制御装置。

【請求項４】　請求項３記載のプロセススイッチ制御装置において，前記スイッチ制御手段(15)は，前記スイッチ抑止状態におけるプロセススイッチの遅延が所定の時間を越えた場合には割り込みを起こし，一つのプロセスが実行権を占有しないように制御する手段を備えたことを特徴とするプロセススイッチ制御装置

【請求項５】　請求項３記載のプロセススイッチ制御装置において，前記スイッチ制御手段(15)は，前記スイッチ抑止状態において非同期な外部割り込みがあった場合に，その割り込みの発生を遅延させ，現在実行中のプロセスの入出力終了またはメッセージ通信終了のイベントが発生してから保留した割り込みを発生させる手段を備えたことを特徴とするプロセススイッチ制御装置。

2

【請求項６】　複数のプロセスが時分割的に動作するマルチプロセッシング・システムにおけるプロセス制御方法において，入出力を実行するプロセス，またはメッセージを送信もしくは受信するプロセスを一つに制限するために，一つのプロセスが入出力を開始またはメッセージの送信もしくは受信を開始してからそれが終了するまでのクリティカルな状態を管理し，そのクリティカルな状態においては他の実行可能なプロセスが入出力を開始またはメッセージの送信もしくは受信を開始することを抑止し，入出力またはメッセージの送信もしくは受信を，マルチプロセッシング環境において複数のプロセスからユーザプログラムのレベルで起動可能とすることを特徴とするプロセス制御方法。

【請求項７】　複数のプロセスが時分割的に動作するマルチプロセッシング・システムにおけるプロセス制御方法において，入出力を実行するプロセスまたはメッセージを送信もしくは受信するプロセスを一つに制限するために，所定のアドレス空間のアクセス権を，入出力を開始したプロセスまたはメッセージの送信もしくは受信を開始したプロセスに与え，他の実行可能なプロセスの前記アドレス空間へのアクセスを禁止し，前記アクセス権を持つプロセスの入出力が終了またはメッセージの送信もしくは受信が終了したときに前記アクセス権を変更し，入出力またはメッセージの送信もしくは受信を，複数のプロセスからユーザプログラムのレベルで起動可能とすることを特徴とするプロセス制御方法。

【発明の詳細な説明】

【０００１】

【産業上の利用分野】本発明は，複数のプロセスが時分割的に動作するマルチプロセッシング・システムにおけるプロセススイッチ制御装置および制御方法に関するものである。

【０００２】近年，計算機システムの高速化が要求されている。このような計算機システムでは，一般に複数のプロセスが走行している。特に，計算性能を上げるために複数のシステムを接続して，相互に通信を行うシステムも用いられており，この場合に利用されるメッセージ通信は，計算性能を犠牲にしないためにもオーバーヘッドができるだけ小さいことが要求される。同様に入出力においてもオーバーヘッドが小さいことが要求される。なお，ここでプロセスとはＣＰＵ実行権を得て走行する計算機システムの実行単位をいい，この実行単位にはいわゆるタスクと呼ばれるものも含まれる

【０００３】

【従来の技術】従来，計算機システムにおいて複数のプロセスが走行している場合，一連の入出力（Ｉ／Ｏ）処理にはオペレーティング・システム（以下，ＯＳという）を介在させる必要があった。しかし，ＯＳを介在さ

せると，ユーザプロセスからＯＳへの制御移行に伴う処理などのために，非常に計算機の処理時間がかかる．このＯＳによるオーバーヘッドを削減するために，ＯＳを介さないユーザレベルの転送方式として，特開平３－１５０６５９号公報（データ転送方式），特開平５－２３３４４０号公報（バッファ機能を備えたデータ転送方式）などに示される技術が知られている。

【０００４】しかし，従来のこれらの方式は，１プロセッサ当たりメッセージを送信できるプロセスが一つである場合に利用できる技術であり，マルチプロセッシング環境では利用するのは難しかった。

【０００５】

【発明が解決しようとする課題】従来，ＯＳを介さずにユーザレベルでメッセージ送信や，入出力操作を可能にしようとした場合，マルチプロセッシング環境では，一つの操作（例えばメッセージ送信）が複数のトランザクションに分割された場合に，トランザクションの間にプロセススイッチが発生して，他のプロセスによる操作が割り込み，一つの操作として意味を持たなくなってしまうという問題があった。

【０００６】図１２は本発明の課題説明図である。図１２（Ａ）は，一般のプロセススイッチを表している。タスクＡが走行しているときに，プロセススイッチを起こす割り込みが入ると，ＯＳがタスクＢにＣＰＵ時間を与えてタスクＢが走り出す。

【０００７】図１２（Ｂ）は，メッセージ送信や入出力をＳｔａｒｔからＥｎｄまでタスクＡが行っている様子を表している。ここで，タスクＡが(1) に示すように入出力を開始する前に，プロセススイッチを起こす割り込みがあった場合には，プロセススイッチによってＣＰＵ実行権をタスクＢに切り替えても何ら問題はない。これに対して，(2) に示すようにメッセージ送信や入出力を，ＯＳを介さずにユーザレベルでタスクＡが行っている最中（critical section）に，プロセススイッチを起こす割り込みが発生すると，問題が発生する。

【０００８】メッセージ送信や入出力がユーザレベルで行われているために，ここで，プロセススイッチによりタスクＢがＣＰＵ時間を与えられて走行を開始すると，タスクＡと同様なメッセージ送信や入出力を行うことがあり，タスクＡの操作と競合が発生して正しい操作が行われなくなる。

【０００９】本発明は上記問題点の解決を図り，マルチプロセッシング環境においても，ＯＳを介さない効率の高いメッセージの送信および入出力を可能とすることを目的とする

【００１０】

【課題を解決するための手段】図１は本発明の原理説明図である　図１（Ａ）において，１０はＣＰＵおよびメモリなどからなる処理装置，１１はマルチプロセッシング環境を提供するオペレーティング・システム（ＯＳ），１２Ａ，１２ＢはＣＰＵ実行権を得て走行する計算機システムの実行単位であるプロセス（以下の説明では，プロセスの例としてタスクを代表させて説明する），１３はプロセスの切り替えを制御するプロセススイッチ制御装置，１４はプロセスの切り替えを制御するための状態を管理する状態管理手段，１５はプロセスを切り替えるタイミング信号であるスイッチ信号(Switch)の出力を制御するスイッチ制御手段，１６はネットワーク等へのメッセージの転送を制御する転送制御装置，１７はディスク装置等のＩ／Ｏデバイスを表す。



【００１１】また，Ｓｔａｒｔは，現在実行中のプロセスが転送制御装置１６によるメッセージ通信またはＩ／Ｏデバイス１７に対する入出力を開始するというイベントを示す信号，Ｅｎｄは，現在実行中のプロセスが転送制御装置１６によるメッセージ通信またはＩ／Ｏデバイス１７に対する入出力を終了するというイベントを示す信号，ｒｅｓｅｔは，状態をクリアすることを指示する信号，Ｓｗｉｔｃｈは，プロセススイッチを起こすことを指示する信号を表す。

【００１２】処理装置１０では，ＯＳ１１の制御のもとに，複数のプロセス（タスクＡ）１２Ａ，プロセス（タスクＢ）１２Ｂ，…が時分割的にＣＰＵ時間を与えられて走行する。本発明では，これらの複数のタスクＡ，Ｂ，…の走行を切り替えるために，状態管理手段１４およびスイッチ制御手段１５を持つプロセススイッチ制御装置１３を有する。

【００１３】請求項１記載の発明では，プロセススイッチ制御装置１３の状態管理手段１４は，少なくとも，プロセススイッチを起こすことのできるフリー状態と，現在実行中のプロセスが入出力中またはメッセージ通信中であるためにプロセススイッチを起こすことを抑止するスイッチ抑止状態とを含む状態を管理する。また，スイッチ制御手段１５は，前記状態に応じてプロセススイッチを起こすタイミングに幅を持たせ，前記スイッチ抑止状態にあるときにはプロセススイッチが起きないようにプロセスのスイッチ信号を制御する。

【００１４】請求項２記載の発明では，状態管理手段１４は，少なくとも，プロセススイッチを起こすことのできるフリー状態と，現在実行中のプロセスが入出力中またはメッセージ通信中であるためにプロセススイッチを起こすことを抑止するスイッチ抑止状態と，プロセススイッチのトリガとなるタイマー・イベントが所定の時間内に発生する状態になったことを示すスイッチ予告促進状態とを含む状態を管理する　また，スイッチ制御手段１５は，前記状態に応じてプロセススイッチを起こすタイミングに幅を持たせ，前記スイッチ抑止状態にあるときにはプロセススイッチが起きないようにプロセスのスイッチ信号を制御するとともに，前記スイッチ予告促進状態において現在実行中のプロセスが入出力中またはメッセージ通信中に入ることを示すイベントが発生したと

きには，そのプロセスが入出力中またはメッセージ通信中に入る前にプロセスのスイッチ信号を出力する。

【００１５】請求項３記載の発明では，さらにスイッチ制御手段１５は，前記スイッチ抑止状態においてプロセススイッチを起こすためのタイマー・イベントまたはその予告イベントが発生した場合に，プロセススイッチを遅延させ，現在実行中のプロセスの入出力終了またはメッセージ通信終了のイベントによりプロセスのスイッチ信号を出力し，プロセススイッチを起こす手段を備える。

【００１６】また，請求項４記載の発明では，スイッチ制御手段１５は，前記スイッチ抑止状態におけるプロセススイッチの遅延が所定の時間を越えた場合には割り込みを起こし，一つのプロセスが長時間にわたって実行権を占有しないように制御する手段を備える。

【００１７】さらにまた，請求項５記載の発明では，スイッチ制御手段１５は，前記スイッチ抑止状態において非同期な外部割り込みがあった場合に，その割り込みの発生を遅延させ，現在実行中のプロセスの入出力終了またはメッセージ通信終了のイベントが発生してから保留した割り込みを発生させる手段を備える。

【００１８】請求項６記載の発明では，入出力を実行するプロセス，またはメッセージを送信もしくは受信するプロセスを一つに制限するために，一つのプロセスが入出力を開始またはメッセージの送信もしくは受信を開始してからそれが終了するまでのクリティカルな状態を管理し，そのクリティカルな状態における他の実行可能なプロセスの入出力起動またはメッセージ送信／受信の開始を抑止することにより，入出力またはメッセージの送信もしくは受信を，マルチプロセッシング環境においてもユーザプログラムのレベルで起動可能とする。

【００１９】請求項７記載の発明では，入出力を実行するプロセス，またはメッセージを送信もしくは受信するプロセスを一つに制限するために，所定のアドレス空間のアクセス権を入出力を開始またはメッセージの送信もしくは受信を開始したプロセスに与え，他の実行可能なプロセスの前記アドレス空間へのアクセスを禁止し，前記アクセス権を持つプロセスの入出力が終了またはメッセージの送信もしくは受信が終了したときに前記アクセス権を変更することにより，入出力またはメッセージの送信もしくは受信を，マルチプロセッシング環境においてもユーザプログラムのレベルで起動可能とする

【００２０】

【作用】本発明では，例えば図１（Ｂ）に示すように，プロセススイッチ（タスクスイッチ）が起こるタイミングに幅を持たせることによって，プロセスが不可分な操作を行っている間に，プロセススイッチが発生することを防ぐので，マルチプロセッシング環境下においてもメッセージ送信／受信，入出力操作をユーザプログラムのレベルで実行することができるようになる



【００２１】プロセススイッチを起こすタイミングの周辺で，メッセージ送信／受信または入出力の起動がかかった場合には，その起動を一時保留し，プロセススイッチを起こすようにするので，プロセススイッチの時間間隔が極端に長くなるのを事前に防ぐことが可能になる。

【００２２】プロセスがメッセージ送信／受信または入出力中である場合には，プロセススイッチを起こすイベントを遅延させ，メッセージ送信／受信または入出力の終了イベントによってプロセススイッチを起こすことにより，送信／受信または入出力途中でプロセススイッチが起きないように制御することができる。

【００２３】さらに，プロセススイッチを起こすイベントの遅延が，ある一定時間を越えた場合には，割り込みを起こすことにより，一つのプロセスがＣＰＵ時間を極端に長い時間占有しないようにすることができる。

【００２４】非同期な割り込みに対しても，プロセスがメッセージ送信／受信または入出力中である場合には，その割り込みの発生を遅延させることにより，プロセスの不可分の処理を保証することが可能になる。

【００２５】また，一つのプロセスが入出力またはメッセージ送信／受信を開始してから，終了するまでの間に，他のアクティブ（実行可能）なプロセスが入出力またはメッセージ送信／受信を開始できないように，プロセスが使用権を持つアドレス空間のアクセス権を制御することにより，入出力またはメッセージ送信／受信を行うプロセスを一つに制限し，マルチプロセッシング環境において，入出力またはメッセージ送信／受信を，複数のプロセスからユーザプログラムのレベルで起動できるようにすることが可能になる。

【００２６】

【実施例】図２ないし図４は本発明の実施例における状態遷移説明図，図５および図６は本発明の実施例におけるプロセススイッチのタイミング説明図，図７は本発明の実施例による非同期割り込みの制御説明図，図８は本発明の実施例におけるアクセス権の制御説明図である。

【００２７】請求項１記載の発明に対応する第１の実施例では，プロセススイッチ制御装置１３は，図２（Ａ）に示すように，ｆｒｅｅ状態およびｕｓｅｄ状態の二つの状態を管理する。ｆｒｅｅ状態は，現在実行中のプロセスが入出力中でもメッセージ送信／受信中でもなく，プロセススイッチを起こすことのできる状態である。ｕｓｅｄ状態は，現在実行中のプロセスが入出力中またはメッセージ送信／受信中であり，プロセススイッチを起こすことを抑止しなければならない状態である。

【００２８】初期化信号ｉｎｉｔｉａｌ（またはリセット信号ｒｅｓｅｔ）のイベントにより，プロセススイッチ制御装置１３はｆｒｅｅ状態になる　ｆｒｅｅ状態において，現在実行中のプロセスが入出力またはメッセージ送信／受信を開始するというイベントを示す信号Ｓｔａｒｔが入ると，許可信号ａｃｋを出力し，ｕｓｅｄ状

態に遷移する　ｕｓｅｄ状態においては，プロセススイッチを発生させるイベントが発生してもプロセススイッチを起こさずに，現在実行中のプロセスが入出力またはメッセージ送信／受信を終了するというイベントを示す信号Ｅｎｄが入ったときに，許可信号ａｃｋを出力するとともに，プロセススイッチを起こすことを指示するスイッチ信号Ｓｗｉｔｃｈを処理装置１０に対して出力し，ｆｒｅｅ状態に遷移する。

【００２９】請求項２記載の発明に対応する第２の実施例では，プロセススイッチ制御装置１３は，図２（Ｂ）に示すように，ｆｒｅｅ状態，ｕｓｅｄ状態およびｓｗｉｔｃｈ状態の三つの状態を管理する。ｆｒｅｅ状態は，現在実行中のプロセスが入出力中でもメッセージ送信／受信中でもなく，プロセススイッチを起こすことのできる状態である。ｕｓｅｄ状態は，現在実行中のプロセスが入出力中またはメッセージ送信／受信中であり，プロセススイッチを起こすことを抑止しなければならない状態である。ｓｗｉｔｃｈ状態は，プロセススイッチのトリガとなるタイマー信号ｔｉｍｅｒが所定の時間内に発生する状態になったことを示すスイッチ予告促進状態である。

【００３０】初期化信号ｉｎｉｔｉａｌ（またはリセット信号ｒｅｓｅｔ）のイベントにより，プロセススイッチ制御装置１３はｆｒｅｅ状態になる。ｆｒｅｅ状態において，現在実行中のプロセスが入出力またはメッセージ送信／受信を開始するというイベントを示す信号Ｓｔａｒｔが入ると，許可信号ａｃｋを出力し，ｕｓｅｄ状態に遷移する。また，ｆｒｅｅ状態において，プロセススイッチのトリガとなるタイマー信号ｔｉｍｅｒが所定の時間内に発生することを示す予告タイマー信号ｔｉｍｅｒ０が入ると，ｓｗｉｔｃｈ状態に遷移する。

【００３１】ｕｓｅｄ状態においては，プロセススイッチを発生させるイベントが発生してもプロセススイッチを起こさずに，現在実行中のプロセスが入出力またはメッセージ送信／受信を終了するというイベントを示す信号Ｅｎｄが入ったときに，許可信号ａｃｋを出力するとともに，プロセススイッチを起こすことを指示するスイッチ信号Ｓｗｉｔｃｈを処理装置１０に対して出力し，ｆｒｅｅ状態に遷移する。

【００３２】ｓｗｉｔｃｈ状態において，プロセススイッチのトリガとなるタイマー信号ｔｉｍｅｒが入ると，スイッチ信号Ｓｗｉｔｃｈを出力してプロセススイッチを起こし，ｆｒｅｅ状態に遷移する。また，現在実行中のプロセスが入出力またはメッセージ送信／受信を開始するというイベントを示す信号Ｓｔａｒｔが入ると，許可信号ａｃｋを出力するとともに，スイッチ信号Ｓｗｉｔｃｈを出力し，他の実行可能なプロセスに強制的にスイッチさせて，ｆｒｅｅ状態に遷移する　また，ｓｗｉｔｃｈ状態でリセット信号ｒｅｓｅｔが入ったときには，許可信号ａｃｋを出力し，ｆｒｅｅ状態に遷移する

【００３３】図５（Ａ）は，この第２の実施例における動作例を示している。タスクＡがｆｒｅｅ状態で走行中に予告タイマー信号ｔｉｍｅｒ０が入ると，ｓｗｉｔｃｈ状態に遷移する。このとき，タスクＡが入出力またはメッセージ送信／受信を開始しようとすると，そのイベントを示す信号Ｓｔａｒｔが検出されるが，ｓｗｉｔｃｈ状態になっているので，タスクスイッチを起こし，タスクＡからＣＰＵ実行権を奪って他の実行可能なタスクＢにＣＰＵ実行権を与える。

【００３４】請求項３記載の発明に対応する第３の実施例では，プロセススイッチ制御装置１３は，図２（Ｃ）に示すように，ｆｒｅｅ状態，ｕｓｅｄ状態，ｓｗｉｔｃｈ状態およびｓｗｉｔｃｈ０状態の四つの状態を管理する。ｆｒｅｅ状態，ｕｓｅｄ状態，ｓｗｉｔｃｈ状態は，前述した図２（Ｂ）に示す状態と同様である。ｓｗｉｔｃｈ０状態は，ｕｓｅｄ状態において，予告タイマー信号ｔｉｍｅｒ０が入ったときに遷移するスイッチ準備状態である。

【００３５】初期化信号ｉｎｉｔｉａｌ（またはリセット信号ｒｅｓｅｔ）のイベントにより，プロセススイッチ制御装置１３はｆｒｅｅ状態になる。ｆｒｅｅ状態において，現在実行中のプロセスが入出力またはメッセージ送信／受信を開始するというイベントを示す信号Ｓｔａｒｔが入ると，許可信号ａｃｋを出力し，ｕｓｅｄ状態に遷移する。また，ｆｒｅｅ状態において，プロセススイッチのトリガとなるタイマー信号ｔｉｍｅｒが所定の時間内に発生することを示す予告タイマー信号ｔｉｍｅｒ０が入ると，ｓｗｉｔｃｈ状態に遷移する。

【００３６】ｕｓｅｄ状態において，現在実行中のプロセスが入出力またはメッセージ送信／受信を終了するというイベントを示す信号Ｅｎｄが入ったときに，許可信号ａｃｋを出力し，ｆｒｅｅ状態に遷移する。一方，ｕｓｅｄ状態において，予告タイマー信号ｔｉｍｅｒ０が入ると，プロセススイッチの準備のためにｓｗｉｔｃｈ０状態に遷移する。

【００３７】ｓｗｉｔｃｈ０状態では，現在実行中のプロセスが入出力またはメッセージ送信／受信を終了するというイベントを示す信号Ｅｎｄが入ると，許可信号ａｃｋを出力するとともに，プロセススイッチを起こすことを指示するスイッチ信号Ｓｗｉｔｃｈを処理装置１０に対して出力し，ｆｒｅｅ状態に遷移する

【００３８】スイッチ予告促進状態であるｓｗｉｔｃｈ状態における状態遷移は，図２（Ｂ）で説明した状態遷移と同様である　図５（Ｂ）は，この第３の実施例における動作例を示している。タスクＡが入出力またはメッセージ送信／受信を開始し，ｕｓｅｄ状態になっているときに，予告タイマー信号ｔｉｍｅｒ０が入ると，ｓｗｉｔｃｈ０状態に遷移する　ｓｗｉｔｃｈ０状態になると，その後のプロセススイッチは遅延され，タスクＡが

入出力またはメッセージ送信／受信を終了したことを示す信号Ｅｎｄによってタスクスイッチを起こし，タスクＢにＣＰＵ実行権が与えられる。

【００３９】請求項４記載の発明に対応する第４の実施例では，プロセススイッチ制御装置１３は，図３（Ａ）に示すように，ｆｒｅｅ状態，ｕｓｅｄ状態，ｓｗｉｔｃｈ状態およびｓｗｉｔｃｈ０状態の四つの状態を管理する。ｆｒｅｅ状態，ｕｓｅｄ状態，ｓｗｉｔｃｈ状態における状態遷移は，図２（Ｃ）に示す状態遷移と同様である。ｓｗｉｔｃｈ０状態についても，図２（Ｃ）の例とほぼ同様であるが，プロセススイッチを遅延させるスイッチ抑止状態（ｕｓｅｄ状態またはｓｗｉｔｃｈ０状態）において，現在実行中のプロセスが長時間にわたってＣＰＵ時間を占有してしまうことを防ぐために，ある一定時間が経過した場合にはＣＰＵに対して割り込み信号Ｉｎｔｅｒｒｕｐｔを出力してｆｒｅｅ状態に戻す。

【００４０】図６（Ａ）は，この第４の実施例における動作例を示している。タスクＡが入出力（Ｉ／Ｏ）を開始している場合には，タスクスイッチをある時間（ＴｉｍｅＯｕｔ）だけ引き延ばし，入出力が終わるのを期待する。その時間内に終わらない場合には，その入出力を中断し，必要に応じてタスクＡに関する処理途中の情報を保存して，タスクＢにタスクスイッチする。

【００４１】請求項５記載の発明に対応する第５の実施例では，プロセススイッチ制御装置１３は，図３（Ｂ）に示すように，ｆｒｅｅ状態，ｕｓｅｄ状態およびｍａｓｋ状態の三つの状態を管理する。ｆｒｅｅ状態，ｕｓｅｄ状態については，前述の例と同様である。ｍａｓｋ状態は，ｕｓｅｄ状態において非同期な外部割り込みＥｘＩｎｔｅｒｒｕｐｔがあった場合に遷移する状態である。非同期な外部割り込みＥｘＩｎｔｅｒｒｕｐｔがあった場合にｕｓｅｄ状態からｍａｓｋ状態に遷移することにより，その非同期な外部割り込みの発生を遅延させ，ある一定の時間が経過したことを示すタイムアウトのイベント（ＴＯ：ｔｉｍｅｏｕｔ），あるいは現在実行中のプロセスの入出力またはメッセージ通信終了のイベントを示す信号Ｅｎｄによって，保留していた割り込みの信号Ｉｎｔｅｒｒｕｐｔを処理装置１０に送出する。

【００４２】図７は，この第５の実施例を実現するための非同期割り込み制御回路の例を示す。従来の通常の非同期割り込みでは，図７（Ａ）に示すように，非同期割り込み等の割り込み信号は，ＯＲ回路６０により論理的にＯＲされ，ＣＰＵに対して与えられる　これに対し，本実施例では，例えば図７（Ｂ）に示すような制御回路が用いられる。現在実行中のタスクが入出力（またはメッセージ通信）を開始するときに，Ｉ／Ｏ　Ｓｔａｒｔの信号によってセット／リセットフリップフロップ６１をセットし，入出力実行中であることを記憶しておく



この状態のときには，セット／リセットフリップフロップ６１の出力であるｍａｓｋの反転信号は０であり，ＯＲ回路６０の出力はＡＮＤ回路６２によってマスクされ，ＣＰＵに割り込みがかかるのが抑止される。これによって，現在実行中のタスクがクリティカルな入出力を行っている最中に，非同期割り込み等が発生するのを防ぐことができる。実行中の入出力が終了すると，Ｉ／Ｏ　Ｅｎｄ信号によってセット／フリップフロップ６１がリセットされ，割り込みのマスクは解除される。

【００４３】請求項６記載の発明に対応する第６の実施例では，プロセススイッチ制御装置１３は，図４（Ａ）に示すように，ｆｒｅｅ状態，ｕｓｅｄ状態の二つの状態を管理する。本実施例では，例えば一つのプロセスがメッセージを送出し始めてから，送出し終わるまでの間に，他のアクティブ（実行可能）なプロセスがメッセージの送出を行えないようにするために，次のように制御する。

【００４４】初期化信号ｉｎｉｔｉａｌ（またはリセットｒｅｓｅｔ）のイベントにより，プロセススイッチ制御装置１３はｆｒｅｅ状態になる。ｆｒｅｅ状態において，現在実行中のプロセスが入出力またはメッセージ送信／受信を開始するというイベントを示す信号Ｓｔａｒｔが入ると，許可信号ａｃｋを出力し，ｕｓｅｄ状態に遷移する。ｕｓｅｄ状態においては，他のアクティブ（実行可能）なプロセスが入出力の起動またはメッセージ通信の開始を指示し，入出力またはメッセージ送信／受信を開始するというイベントを示す信号Ｓｔａｒｔを出力しても，不許可信号ｎａｃｋが返される。そして，現在実行中のプロセスが入出力またはメッセージ送信／受信を終了するというイベントを示す信号Ｅｎｄが入ったときに，許可信号ａｃｋを出力するとともに，プロセススイッチを起こすことを指示するスイッチ信号Ｓｗｉｔｃｈを処理装置１０に対して出力し，ｆｒｅｅ状態に遷移する。

【００４５】図６（Ｂ）は，この第６の実施例における動作例を示している。タスクＡ，タスクＢ，タスクＣが走行可能状態であったとする。本実施例は，例えばタスクＡが入出力（Ｉ／Ｏ）処理を開始した状態で，タスクＢやタスクＣにＣＰＵ実行権が切り替わった場合には，タスクＢやタスクＣに対してＩ／Ｏにアクセスする権利を与えないようにするものである。図６（Ｂ）に示す動作例では，タスクＡが入出力処理を終えた後に，タスクＢが入出力処理を開始している。

【００４６】請求項７記載の発明に対応する第７の実施例では，例えば図８（Ａ）に示すように，タスクＡ，タスクＢ，タスクＣが走行可能状態にある場合に，入出力（Ｉ／Ｏ）を起動できるアドレス空間（User accessible I/O）を，タスクＡのみに解放して，他のタスクＢ，Ｃからはこのアドレス空間にアクセスできないようにアクセス権を制限する　アクセス権の制限は，例えば主記憶

保護などの周知の技術を用いて実現することができる。

【００４７】さらに、この第７の実施例において、アクセス権が制限されているタスクＢ、ＣがＩ／Ｏをアクセスしようとすると、例えば割り込みを発生させてＯＳに制御を移し、ＯＳは割り込みを起こしたタスクＢまたはタスクＣの状態を変化させ（Ｉ／Ｏ待ちなど）、タスクＡのＩ／Ｏが終了するまで、Ｉ／Ｏ待ちなどの状態にあるタスクＢ、ＣにＣＰＵ時間を与えないようにする。図４（Ｂ）はその例における状態遷移を示している。タスクＡのＩ／Ｏの開始により、ｆｒｅｅ状態からｕｓｅｄ状態に遷移する。このｕｓｅｄ状態において、他のタスクＢ、ＣがＩ／Ｏを起動しようとしても、タスクスイッチによってＩ／Ｏ待ちになる。タスクＡのＩ／Ｏが終了するとｆｒｅｅ状態に戻り、他のタスクＢ、ＣのＩ／Ｏの起動が可能になる。

【００４８】図７（Ｂ）は、この第７の実施例を実現するためのシステム構成の例を示している。図中，５０はＣＰＵ、５１はメモリ制御装置（ＭＭＵ）、５２はメモリを表す。メモリ制御装置５１は，ＣＰＵ５０が出すアドレスをチェックする機構を持つ。ＯＳは、どのタスクがＩ／Ｏ領域にアクセスできるかをメモリ制御装置５１のモジュールに設定しておく。これにより，アクセス権のないタスクが発行したアドレスが不適切である場合には，割り込み（Ｉｎｔｅｒｒｕｐｔ）を発生させ，タスクスイッチを起こす。メモリ制御装置５１によるアクセス保護機能の利用はマルチタスクの環境では一般的であり，本実施例を実現するための装置の変更はほとんど必要がない。もちろん、これと同等な機能を持つアクセスチェック機構によって本実施例を実現することも可能である。

【００４９】図９（Ａ）は，本発明の実施例によるプロセススイッチ制御装置の構成例を示す図であり，図９（Ｂ）はプロセススイッチ制御装置におけるカウンタの出力説明図である。また，図１０および図１１は，プロセススイッチ制御装置におけるステートマシンの実現方法説明図である。

【００５０】以下では，請求項３記載の発明の一実施例であるプロセス制御装置の構成例について説明する。図９（Ａ）において，９０はステートマシン（ＦＳＭ）であって，有限な状態を管理し入力信号によって状態を遷移させるとともに，そのときの状態と入力信号とに応じて各種の制御信号を出力する装置である　９１はＣＰＵが出すアドレスをデコードするデコーダ（ＤＥＣ），９２はＣＰＵのクロックｃｌｋをカウントし，所定の値になったときに予告タイマー信号ｔｉｍｅｒ０を出力する第１のカウンタ（ｃｏｕｎｔｅｒ０），９３はクロックｃｌｋをカウントし，所定の値になったときにプロセススイッチのトリガとするタイマー信号ｔｉｍｅｒを出力する第２のカウンタ（ｃｏｕｎｔｅｒ）を表す

【００５１】ＣＰＵが起動するＩ／Ｏやメッセージ送出命令は，この例では，ＣＰＵが出すアドレスによって検出される　デコーダ９１は，そのようなアドレスをデコードすることによって，ＣＰＵのクリティカルな操作の開始時（Ｉ／Ｏの開始，メッセージ送出の開始等）に信号Ｓｔａｒｔを生成し，クリティカルな操作の終了時に信号Ｅｎｄを生成する　同様に，リセット信号ｒｅｓｅｔもＣＰＵが出す特定のアドレスによって生成する。

【００５２】また，第１のカウンタ９２および第２のカウンタ９３は，ＣＰＵのクロックｃｌｋを分周し設定された時間に，それぞれ予告タイマー信号ｔｉｍｅｒ０およびタイマー信号ｔｉｍｅｒをアクティブにする。このカウンタ出力信号のタイミングは，例えば図９（Ｂ）に示すようになっており，タイマー信号ｔｉｍｅｒがアクティブになる所定の時間前に，予告タイマー信号ｔｉｍｅｒ０がアクティブになるように設定されている。タイマー信号ｔｉｍｅｒがアクティブになる時間間隔は，通常の場合におけるプロセススイッチの時間間隔に一致し，この時間はＣＰＵのクロックｃｌｋに対して十分に長い時間である。

【００５３】これらの信号はステートマシン９０に入力され，ステートマシン９０によって，例えば図２（Ｃ）に示すような状態遷移に応じてａｃｋ（ｎａｃｋ），Ｓｗｉｔｃｈの信号が生成される。ＣＰＵは許可信号ａｃｋを受け取ると正常にバストランザクションが終了（成功）したことを認識する。許可信号ａｃｋが出力されなかった（または不許可信号ｎａｃｋが出力された）場合，ＣＰＵはアクセスが正常に終了していないことを認識して，リトライを行うか停止する。また，スイッチ信号Ｓｗｉｔｃｈによって，他のプロセスを実行するためのプロセススイッチを起こす。

【００５４】以上のようなステートマシン９０は，図１０および図１１に示すようなハードウェア記述言語をコンパイルすることによって，容易に実現することができる。図１０および図１１は，ハードウェア記述言語の一つとして周知のＶｅｒｉｌｏｇを用いて第４の実施例における状態遷移を実現するものを示している。Ｖｅｒｉｌｏｇはよく知られた言語であるので詳しい説明を省略し，ここでは図１０および図１１による表現の概要だけを簡単に説明する。

【００５５】第２行目は，ステートマシンのモジュールの定義文である。第３行目は，入力信号を示す。ここでｉｎｔは，図２においてｉｎｉｔｉａｌとして示されている初期化信号である　第４行目は，出力信号を示す

【００５６】第６行目は，現在の状態ｃｓｔａｔｅと次の状態ｎｓｔａｔｅを管理するレジスタを定義している　第８行目は，２ビットでｆｒｅｅ状態（ＦＲＥＥ＝“００”），ｕｓｅｄ状態（ＵＳＥＤ＝“０１”），ｓｗｉｔｃｈ０状態（ＳＷＩＴＣＨ０＝“１０”），ｓｗｉｔｃｈ状態（ＳＷＩＴＣＨ＝“１１”）の状態を表すことを定義している　これらの状態管理は２個のフリ

ップフロップで実現される。

【００５７】第９行目から第３８行目は，ｃｓｔａｔｅ，ｅｎｄ，ｓｔａｒｔ，ｒｅｓｅｔ，ｔｉｍｅｒ，ｔｉｍｅｒ０またはｉｎｔにより，次の状態がどのように遷移するかを示している。

【００５８】また，第４０行目から最終行までは，ａｃｋおよびｓｗｉｔｃｈの信号を出力する条件を示している。ここでは，第４の実施例を代表させて，図９（Ａ）に示すステートマシン９０を実現するＶｅｒｉｌｏｇ記述の例を示したが，他の実施例におけるステートマシンの状態遷移についても，以上説明した例から推測して当業者であれば容易に実現可能であるので，個々の実現例のこれ以上の詳しい説明は省略する。図１０および図１１に示すＶｅｒｉｌｏｇ記述をコンパイルすることによって，図９（Ａ）に示すステートマシン９０を実現する論理回路を自動設計する技術は確立されている。

【００５９】

【発明の効果】以上説明したように，本発明によれば，ユーザレベルでのメッセージの送出や，Ｉ／Ｏの実現におけるプロセススイッチに関する問題を解決することによって，マルチプロセッシング環境下でも，ユーザレベルで直接メッセージの送出やＩ／Ｏを制御することができ，低いオーバーヘッドでこれらを実現することが可能になる。したがって，計算機の性能を効率よく引き出すことが可能となる。特に，プロセッサ間のメッセージ通信が頻繁に現れるような並列計算機システム等においては性能向上の効果が大きい。

【図面の簡単な説明】

【図１】本発明の原理説明図である。

【図２】本発明の実施例における状態遷移説明図である。

【図３】本発明の実施例における状態遷移説明図である。

【図４】本発明の実施例における状態遷移説明図である。

【図５】本発明の実施例におけるプロセススイッチのタイミング説明図である。

【図６】本発明の実施例におけるプロセススイッチのタイミング説明図である。

【図７】本発明の実施例による非同期割り込みの制御説明図である。

【図８】本発明の実施例におけるアクセス権の制御説明図である。

【図９】本発明の実施例によるプロセススイッチ制御装置の構成例を示す図である。

【図１０】本発明の実施例によるプロセススイッチ制御装置におけるステートマシンの実現方法説明図である。

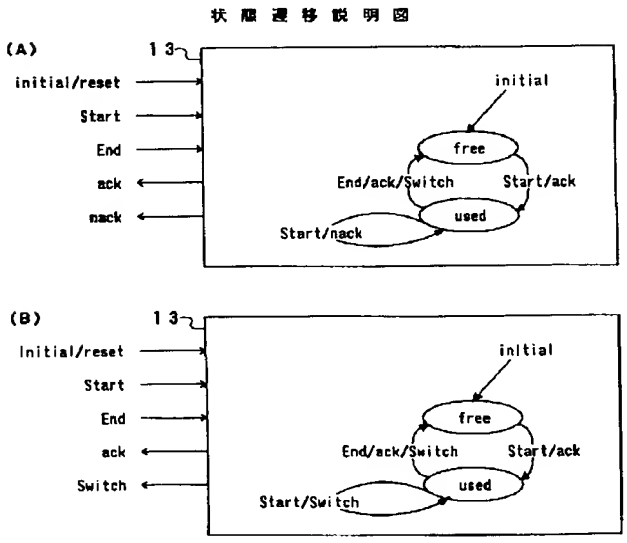
【図１１】本発明の実施例によるプロセススイッチ制御装置におけるステートマシンの実現方法説明図である。

【図１２】本発明の課題説明図である。

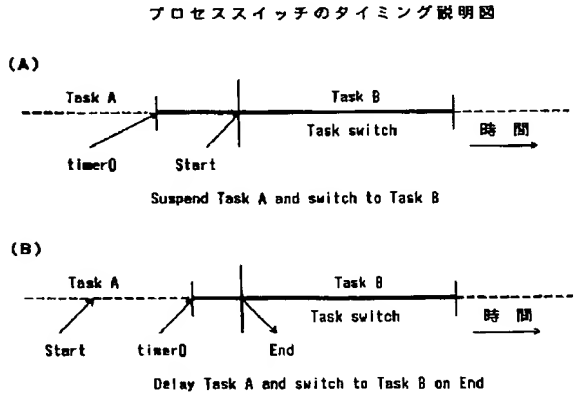
【符号の説明】

１０　処理装置
１１　オペレーティング・システム（ＯＳ）
１２Ａ，１２Ｂ　プロセス
１３　プロセススイッチ制御装置
１４　状態管理手段
１５　スイッチ制御手段
１６　転送制御装置
１７　Ｉ／Ｏデバイス

【図４】

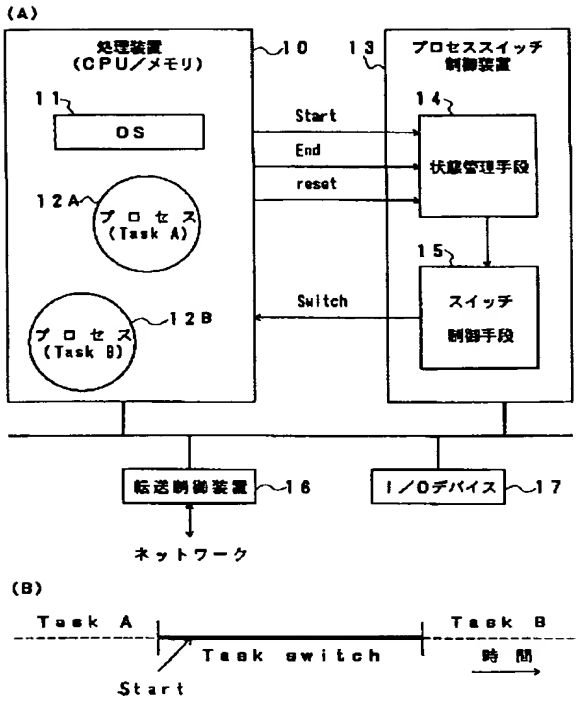



【図５】



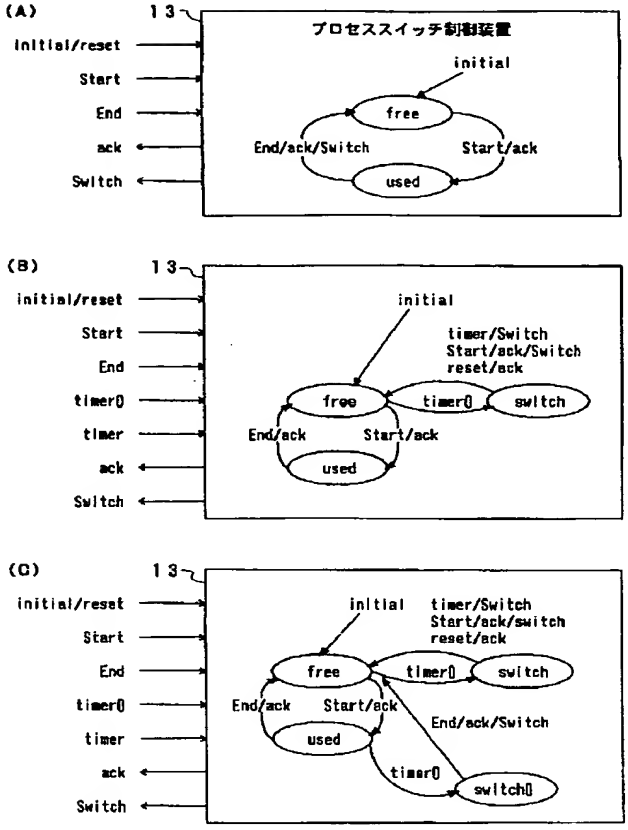
【図１】

本発明の原理説明図



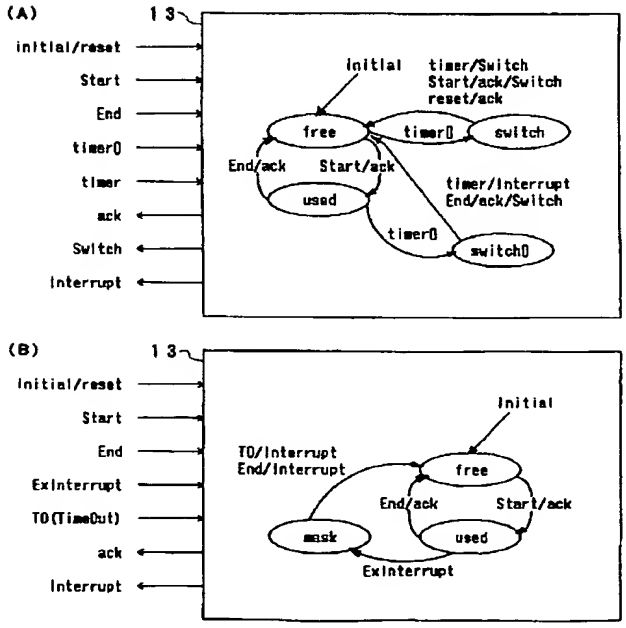

【図２】

状態遷移説明図



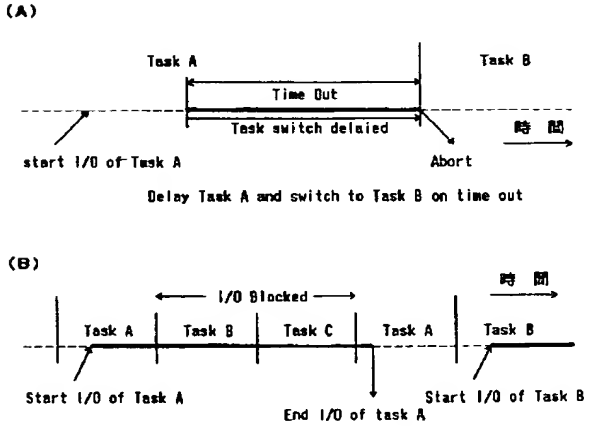

【図３】

状態遷移説明図



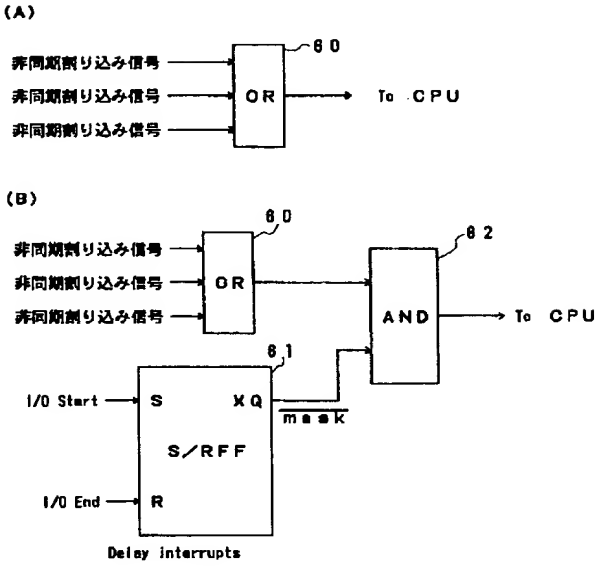

【図６】

プロセススイッチのタイミング説明図



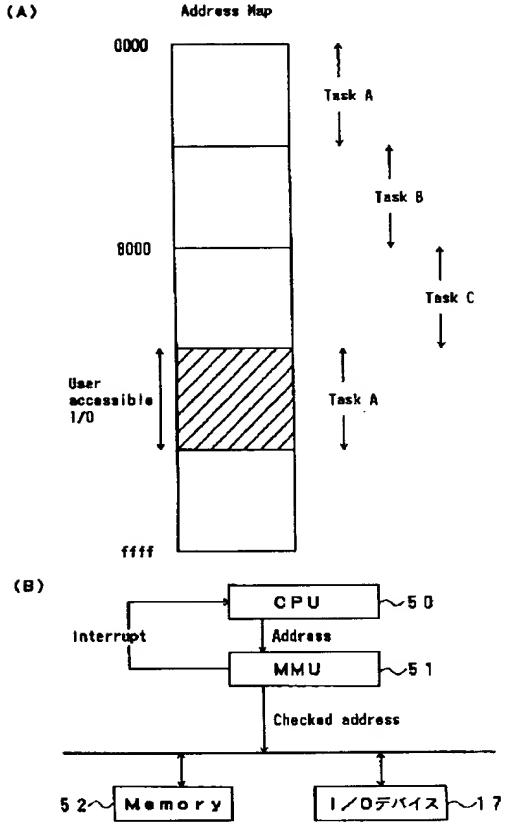
【図７】

非同期割り込みの制御説明図



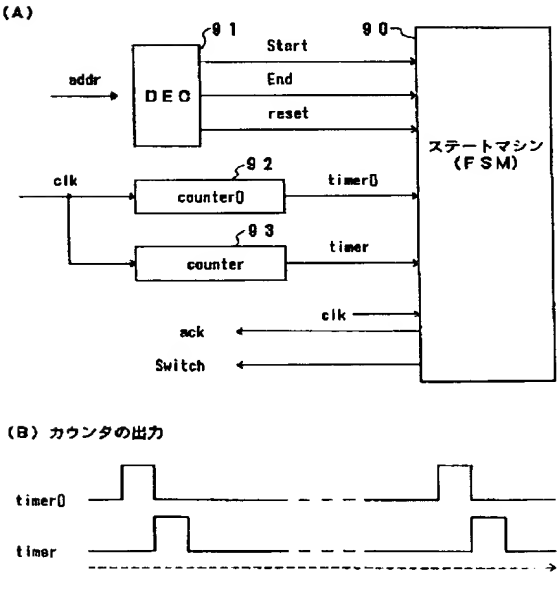

【図８】

アクセス権の制御説明図



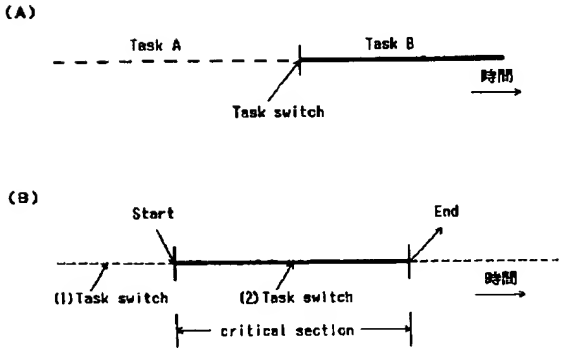

【図９】

プロセススイッチ制御装置の構成例




【図１２】

本発明の課題説明図



【図１０】

ステートマシンの実現方法

```
1  module fsm(end,start,reset,timer,timer0,clk,int,ack,switch);
2  input end,start,reset,timer,timer0,int,clk;
3  output ack,switch;
4
5  reg [1:0] cstate,nstate;
6
7  parameter[1:0] FREE = 2'b00, USED = 2'b01, SWITCH0 = 2'b10, SWITCH = 2'b11;
8  always @ (cstate or end or start
9      or reset or timer or timer0 or int)
10 begin
11    nstate = cstate;
12    if (int)
13       nstate = FREE;
14    else case (cstate)
15       FREE:  if (start)
16                 nstate = USED;
17       USED:  if (end)
18                 nstate = FREE;
19              else if (timer0)
20                 nstate = SWITCH;
   ....
```

【図１１】

ステートマシンの実現方法

```
   ...
21       USED:  if (end)
22                 nstate = FREE;
23              else if (timer0)
24                 nstate = SWITCH0;
25       SWITCH0:
26                 if (end || timer)
27                    nstate = FREE;
28       SWITCH:
29                 if (reset || start || timer)
30                    nstate = FREE;
31       endcase
32    end
33
34    always @ (posedge clk)
35    begin
36       cstate = nstate;
37    end
38
39    wire ack = (cstate == FREE) && start
40             || (cstate == USED) && end
41             || (cstate == SWITCH0) && end;
42             || (cstate == SWITCH) && (reset || start);
43
44    wire switch = (cstate == SWITCH) && (start || timer)
45                || (cstate == SWITCH0) && end;
46
47    endmodule
48
49
```